Software Manual

TRICONTINENT
by Gardner Denver

Valve Controller

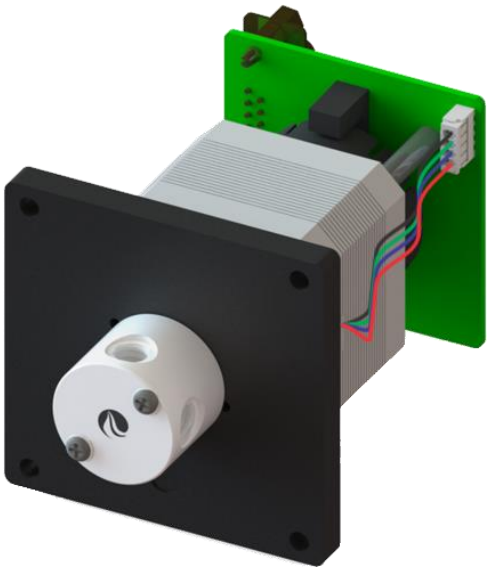


Table of Contents

Overview

The Valve Controller Module can be used to control any of the rotary valves offered by TriContinent. The controller can be used independently, or with C-Series Syringe Pumps for various fluidic applications.

Valve Controller Features

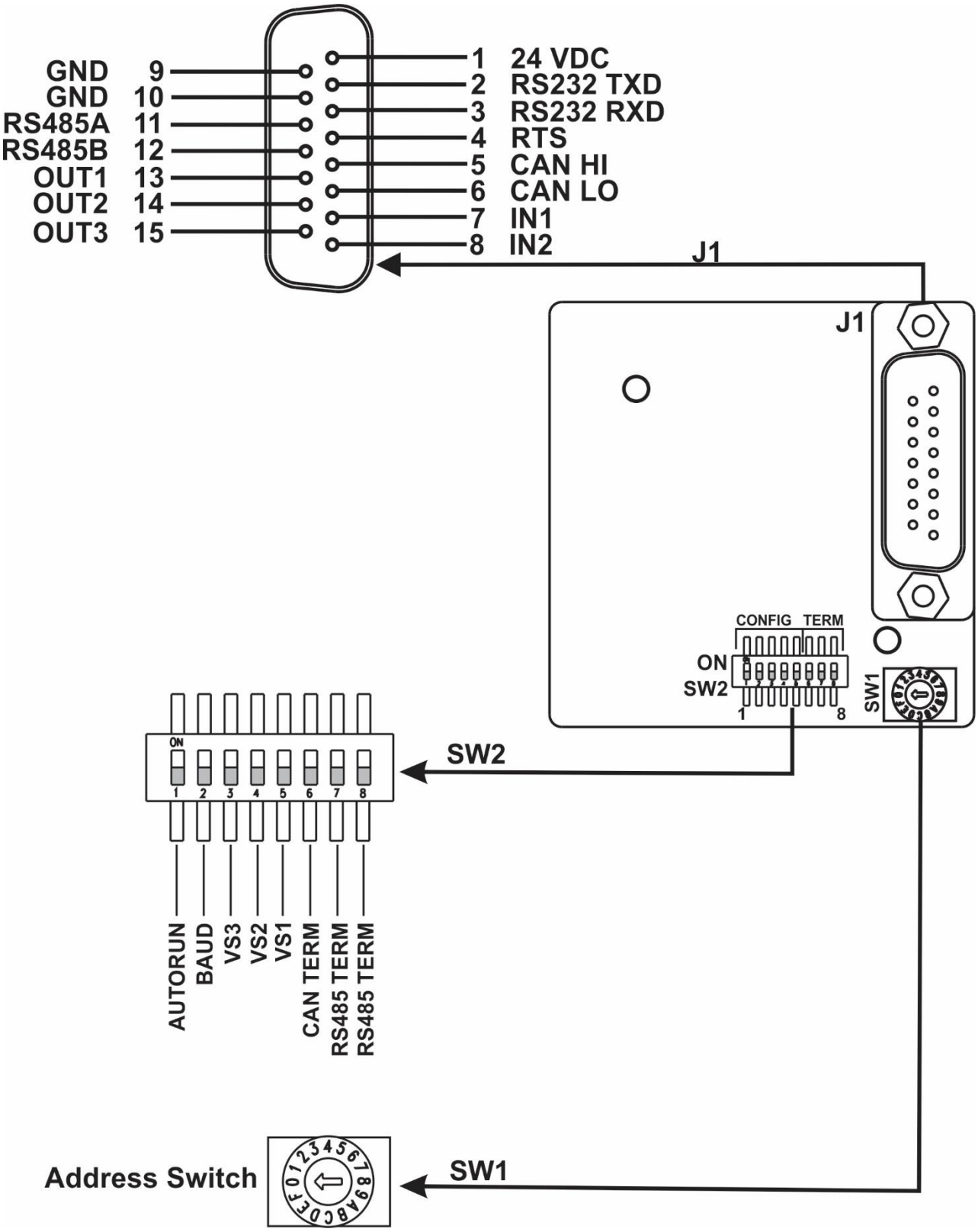
- RS232 & RS485 interface, up to 15 modules can be individually addressed
- CAN bus interface
- C-Series compatible command interface
- Runs off a single +24V supply
- Auxiliary TTL I/O
- EEPROM configurable

Valve Selection Guide

NOTE: The valve type can also be configured using DIP switch settings, see "Valve Selection, DIP Switch Settings (SW2)" table below. The EEPROM configurations below are valid if the valve selection switches are set to EEPROM mode (default).

Valve Type	**CAT Number**	**EEPROM Configuration Command**	**Figure**	**Comments**
120°, 3-Port "Y" Valve	10020 10021 10022 10023	U1	1	
90°, 4-Port	10040 10041 10042 10043	U2	2	
4-Port Distribution Valve	10062 10063	U11	5	Use with [I]<n> and [O]<n> commands
		U4	4	Use with [I],[O],[B] and [E] commands
90°, 3-Port & 4-Port "T" Valve	10115 10025	U5	3	
90°, 4-Port Loop Valve	Call for Availability	U9	6	
7-Port Distribution Valve	Call for Availability	U7	7	
NOTE: EEPROM commands require power to be cycled before taking effect. EEPROM commands are non-volatile so they also only need to be issued once.				

Valve Controller Hardware Information



Configuration Options, DIP Switch Settings (SW2)

Switch #	Description	ON (Default)	OFF
SW2 - 1	AutoRun	Normal operation	Enabled, run stored string on power up
SW2 - 2	Baud	Baud = 9600	Baud = 38400

Valve Configuration Selection, DIP Switch Settings (SW2)

SW2-3 (VS3)	SW2-4 (VS2)	SW2-5 (VS1)	Valve Type
ON	ON	ON	EEPROM Control, Default (U4)
ON	ON	OFF	120°, 3-Port “Y” Valve
ON	OFF	ON	90°, 4-Port
ON	OFF	OFF	90°, 3-Port & 4-Port “T” Valve
OFF	ON	ON	90°, 4-Port Loop Valve
OFF	ON	OFF	4-Port Distribution Valve I,O,BE
OFF	OFF	ON	4-Port Distribution Valve I[n],O[n]
OFF	OFF	OFF	7-Port Distribution Valve

Bus Termination, DIP Switch Settings (SW2)

Switch #	Description	ON (Default)	OFF
SW2 - 6	CAN	Terminated	Unterminated
SW2 - 7	RS485A	Terminated	Unterminated
SW2 - 8	RS485B	Terminated	Unterminated

NOTE: Only the last Device in the chain should be terminated. All other devices should be unterminated, with switches 6-8 set to the OFF position.

Valve Controller Communication Interface

The TCS Valve Controller has three separate communication interfaces: RS232, RS485 and CAN.

RS232 and RS485 are identical from a software standpoint. The main difference between the two is that RS485 can communicate to more than one Valve Controller. RS232 is limited to a single controller communication. RS485 is sometimes referred to as multi-drop, whereas RS232 is single drop.

The TCS Valve Controller has a built in RS232 to RS485 converter. This allows for an RS232 connection from the host computer to the first Valve Controller, and RS485 connections to any additional devices.

CAN communication is completely different in both the hardware layer and software protocols from RS232/RS485. Please refer to CAN bus section for more detail.

Valve Controller Addressing Scheme

As part of the communication protocol, an address for each device must be specified. The user has the option of addressing a single device, two devices (dual device), four devices (quad device), or all 15 devices (all devices), depending on the address byte used. Each physical address on the address switch corresponds to a hexadecimal value, as shown in Table 1.

Table 1. Hexadecimal Addressing Scheme

Address (hex)		
RS232/RS485	**CAN**	**Device**
30	0	Master Address (master controller, personal computer, etc)
31..3F	1..F	Addresses single device
41..4F	N/A	Addresses two devices at a time (dual device)
51..5D	N/A	Addresses four devices at a time (quad device)
5F	N/A	Addresses all devices on the bus

For example, a Valve Controller with address switch set to 0 is addressed as device “31h” in the RS232 or RS485 communication protocol, hardware address 1 is addressed as device “32h”, and so on.

Table 2 shows the different address switch settings for each of these configurations.

Table 2. Address Switch Settings in Hex (ASCII)

Switch Setting	Single Device		Dual Device		Quad Device		All Devices	
	Hex Address	ASCII Address	Hex Address	ASCII Address	Hex Address	ASCII Address	Hex Address	ASCII Address
0 1	31 32	1 2	41	A	51	Q	5F	-
2 3	33 34	3 4	43	C				
4 5	35 36	5 6	45	E	55	U		
6 7	37 38	7 8	47	G				
8 9	39 3A	9 :	49	I	59	Y		
A B	3B 3C	; <	4B	K				
C D	3D 3E	= >	4D	M	5D	]		
E	3F	?	4F	O				
F	Self-Test							

The user can communicate with all devices in the chain by using address “5Fh”, for example, to initialize all devices at once. Each device can then be controlled independently by using addresses “31h” to 3Fh.”

NOTE: Multiple address commands cannot be used to determine device status, nor will they respond to Report commands. Each device must be queried separately.

Communication Protocols

Three communication protocols are available:

- OEM protocol
- Data Terminal (DT) protocol
- CAN protocol (optional)

NOTE: The Valve Controller firmware automatically detects the OEM, DT or CAN communication protocol. The Baud rate (9600 or 38400) is not automatically detected and needs to be set correctly.

The DT (Data Terminal) protocol is designed to be used with a Terminal emulator program.
TCS Commander Software can also be used to communicate in either DT or OEM mode.

NOTE: If not using a Terminal emulator program, TriContinent recommends using the OEM protocol. It provides increased error checking and recovery.

OEM Communication Protocol

OEM communication is a robust protocol that includes automatic recovery from transmission errors. Table 3 describes each setting within the OEM communication protocol.

Table 3. OEM Protocol Communication Details

Parameter	**Setting**
Character Format	
Baud Rate	9600 or 38400
Data Bits	8
Parity	None
Stop Bit	1
Command Block (see “OEM Protocol Command Block Characters” for details)	
1	STX (02h)
2	Device Address
3	Sequence Number
3+n	Data Block (length n)
4+n	ETX (03h)
5+n	Checksum
Answer Block (see “OEM Protocol Answer Block Characters” for details)	
1	Line Synchronization (FFh)
2	STX (02h)
3	Master Address (0 or 30h)
4	Status Code
4+n	Data block (length n bytes)
5+n	ETX (03h)
6+n	Checksum

OEM Protocol Command Block Characters

The command block characters in the OEM communication protocol are described below. All characters outside the command block are ignored.

When developing a parsing algorithm, the programmer should key on the STX as the beginning of the answer block and the checksum (character after ETX) as the end of the answer block.

STX (02h)

The STX character indicates the beginning of a command string.

Device Address

The device address is specific, set by the rotary switch, for each individual device.

NOTE: The device’s address is the rotary switch setting plus one. For example, if the switch is set to 0, the device’s address is 1 or “31h”.

Sequence Number/Repeat Flag

The sequence number is a single byte that conveys both a sequence number (legal values: 0 to 7) and a bit-flag indicating that the command block is being repeated due to a communications breakdown. The sequence number is used as an identity stamp for each command block. Since it is only necessary that every message carry a different sequence number from the previous message (except when repeated), the sequence number may be toggled between two different values (e.g., "1" and "2") as each command block is constructed. During normal communication exchanges, the sequence number is ignored. If, however, the repeat flag is set, the controller compares the sequence number with that of the previously received command block to determine if the command should be executed or merely acknowledged without executing.

The following two scenarios clarify this error detection mechanism.

Scenario 1

1. The host computer sends a command block stamped with sequence #1 to the device.
2. The device receives the command, sends an acknowledgement to the host, and executes it.
3. Transmission of the acknowledgement message is imperfect; the host does not receive it.
4. The host waits 100 ms for the acknowledgement then retransmits the command block with the sequence number left at 1 and the repeat bit set to indicate a retransmission.
5. The device receives the transmission, identified as such by the repeat bit.
6. The device checks the sequence number against that of the previously received command block. Noting a match, the device sends an acknowledgement to the host, but does not execute the command (since it has already been executed).
7. The host receives the acknowledgment and continues with normal communications.
8. The next command block is stamped with sequence #2 to indicate a new command.

Scenario 2

1. The computer sends a command block stamped with sequence #1 to the device.
2. The device never receives the command due to a communication error and thus does not send an acknowledgement to the host.
3. The host waits 100 ms for the acknowledgement then retransmits the command block with the sequence number left at 1 and the repeat bit set to indicate a retransmission.
4. The device receives the retransmission, identified as such by the repeat bit.
5. The device checks the sequence number against that of the previously received command block. Noting a mismatch, the device recognizes this as a new command block and sends an acknowledgement to the host. It then executes the command.
6. The host receives the acknowledgement and continues with normal communications.
7. The next command block is stamped with sequence #2 to indicate a new command.

The sequence number/repeat byte is constructed as follows:

Bit #	7	6	5	4	3	2	1	0
Value	0	0	1	1	REP	SQ2	SQ1	SQ0

REP: 0 for non-repeated, 1 for repeated

SQ0 – SQ2: sequence value, as follows:

Sequence Value	**SQ2**	**SQ1**	**SQ0**
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Data Block (length n bytes)

The data block consists of the data or commands sent to the device or host (this is an ASCII string). The [Q] command has a data block length of 0 (i.e. no data string exists).

ETX (03h)

The ETX character indicates the end of the command string.

Checksum

The checksum is the last byte of the message string. All bytes (excluding the checksum) are XORed to form an 8-bit checksum. This is appended as the last character of the block. The receiver compares the transmitted value to the computed value. If the two values match, an error free transmission is assumed, otherwise a transmission error is assumed.

OEM Protocol Answer Block Characters

The answer block characters in the OEM communication protocol are described below.

Line synchronization (FFh)

Sent for compatibility reasons. This character does NOT indicate the beginning of a Data Block.

STX (02h)

The STX character indicates the beginning of a response string.

Master Address

The master address is the address of the host system. This should always be “30h” (ASCII “0”).

Status and Error Codes

The status and error codes define device status and signal error conditions. For a description of status and error codes, see “Error Codes and Query Status”.

Data Block (length n bytes)

This is the response from all report commands with the exception of the [Q] command.

ETX (03h)

The ETX character indicates the end of the response string.

Checksum

The checksum is the last byte of the message string. All bytes (excluding line synchronization and checksum) are XORed to form an 8-bit checksum. This is appended as the last character of the block. The receiver compares the transmitted value to the computed value. If the two values match, an error free transmission is assumed, otherwise a transmission error is assumed.

Data Terminal (DT) Protocol

The DT Protocol can be used easily from any terminal or terminal emulator capable of generating ASCII characters at 9600 or 38400 baud, 8 bits, and no parity.

Table 4. DT Protocol Communication Details

Character Format	
Baud Rate	9600 or 38400
Data Bits	8
Parity	None
Stop Bit	1
Command Block (see "DT Protocol Command Block Characters" for details)	
1	Start Character (ASCII "/" or 2Fh)
2	Device Address
2+n	Data Block (length n)
3+n	End Character (Carriage Return ([CR] or 0Dh)
Answer Block (see "DT Protocol Command Block Characters" for details)	
1	Start Answer (ASCII "/" or 2Fh)
2	Master Address (ASCII "0" or 30h)
3	Status Character
3+n	Data block (if applicable)
4+n	ETX (03h)
5+n	Carriage Return (0Dh)
6+n	Line Feed (0Ah)

DT Protocol Command Block Characters

The command block characters in the DT communication protocol are described below.

Start Character

The start character (ASCII "/" or "2Fh") indicates the beginning of a message.

Device Address

The device address is an ASCII character specific to each device.

NOTE: The device's address is the rotary switch setting plus one. For example, if the switch is set to 0, the device's address is 1 or "31h".

Data Block (length n)

The data block consists of the ASCII data or commands sent to the device or host.

End Character

The end character ("0Dh") indicates the end of the message.

DT Protocol Answer Block Characters

The answer block characters comprising the DT communication protocol are described below.

Start Answer

The start character (ASCII “/” or “2Fh”) indicates the beginning of the response.

Master Address

The master address is the address of the host system. This should always be “30h” (ASCII “0”).

Status Character

The status and error codes define device status and signal error conditions. See the description of the [Q] command in “Error Codes and Query Status.”

Data Block

This is the response from all report commands with the exception of the [Q] command.

ETX (03h)

The ETX character indicates the end of the response string.

Carriage Return (0Dh) and Line Feed (0Ah)

These characters terminate the reply block.

CAN Interface Communications

CAN (Controller Area Network) is a two-wire, serial communication bus. It eliminates polling sequences that verify task completion. Using CAN, the devices asynchronously report to the host when they have finished the current task.

NOTE: The TriContinent Valve Controller uses CAN controller chips compatible with Philips Semiconductor CAN Bus specification, version 2.0.

CAN Messages

CAN messages consist of frames. Each frame has an 11-bit Message Identifier (MID) and a 4-bit length identifier. The bits:

- Indicate to which device on the bus the message is directed.
- Identify the message type.
- Show the direction of the message (to or from the host device).
- Represent the length of the data block. Data blocks can be from zero to eight bytes in length. Any message that requires more than eight bytes must be sent in a series of multi-frame messages. The receiving unit then assembles the separate frames into one long string.

CAN Message Construction

Each message frame begins with the MID. The data block (up to 8 bytes in length) follows the MID and length information. The MID makes up three nibbles that are transmitted first in a message frame. The bits are grouped as shown:

Dir

This is the direction bit. It lets the devices on the bus know whether the current message is to or from the host. “0” means that the message is from host to device; “1” means the message is from the device to the host.

NOTE: Peer-to-peer messaging is not supported.

Group

This is the group number (0 – 7). Each type device on the CAN Bus has a group assignment. The Valve Controller is assigned to group 2. The group number “1” is reserved for the boot request procedure.

Device

This is the address of the module in the particular group. Each group can have up to 16 devices. The address value is 0 – 15.

Frame Type

This lets the controller know what type message is coming. See “CAN Frame Types.”

RTR

This bit is not used in TriContinent’s CAN implementation and should always be set to 0.

Data Length Code (DLC)

This is the length of the data block in the message. Data blocks can be from zero to eight bytes in length.

CAN Frame Types

The frame types allow each controller to know what type of command is coming in and enables faster processing of commands. Controllers respond to the frame types described below.

"On-the-Fly" Commands (T)

Normal commands use a frame type zero of "1" (i.e., "Action Commands"). Since commands sent over the CAN Bus with a particular frame type must complete before a subsequent command using the same frame type can be issued, a different ID must be used when issuing an "on-the-fly" command. For this reason, "on-the-fly" commands must be issued over the CAN Bus with a frame type of "0" (zero).

When using "on-the-fly" commands, the frame type "0" commands will not generate completion messages and thus no pairing code is needed (these commands are simply acknowledged immediately).

Action Frames, Type 1

This frame type is used for action commands, such as Initialization commands, Valve commands, or to set operating parameters. All "task-type" commands are sent in this type message frame. When multi-frame messages are used to send an action command, this frame is the end message sent to the controller.

Common Commands, Type 2

This frame is used for commands that are common to every controller on the bus. The frame type is set to “2” and the command is a single ASCII character in the data block. The single ASCII character is described below.

Command	Description	Equivalent DT or OEM Command
0	Reset mode. This resets the controller and begins the boot request procedure.	N/A
1	Start loaded command.	[R]
2	Clear loaded command. This clears out the command buffer.	N/A
3	Repeat last command.	[X]
4	Stop action immediately.	[T]

Multi-Frame Start Message, Type 3

This frame type lets the device know that the next message will be longer than the 8-byte maximum for each frame. Subsequent frames will follow to complete the message.

Multi-Frame Data, Type 4

This frame type is used to identify a frame in the middle of a multi-frame message. The last frame of a multi-frame message for action commands must be type “1”. The last frame of a multi-frame message response from the controller for report commands will be type “6”.

NOTE: There is no type “5” frame

Report/Answer Commands, Type 6

This frame type is used to get information back from the controller. It is similar in operation to the Query commands (i.e. [?]) used in the OEM and DT protocols. The report command is one byte long and is a single ASCII character in the data block.

When the controller responds to a query, the first byte of the data block is the status byte. It is defined like the status byte in the OEM and DT protocols. The next byte is a null character. The remaining six bytes are for the response in ASCII. If the controller is only reporting current status, the message is only two bytes long. If the reply consists of more than six bytes, multi-frame messages are used.

CAN Data Block

The data block tells the device what to do. Controller commands are sent in ASCII just like in OEM or DT. For command strings that are more than eight bytes in length, multi-frame messages are used. This permits long program strings to be sent as with other communications interfaces.

Handling of Device Boot Requests

When the controller is first powered up or receives a system reset command (frame type “2”, command 0), the controller notifies the host of this condition by sending a boot request message at 10 to 12 second intervals until it receives a proper response. The group number is “1” for the boot request message. The frame type is “2” when the controller sends messages to the host, and the frame type must be “0” when the host replies to the boot request.

Example 1. The controller is set to address 0

Controller sends:

Dir	Group	Device	Frame Type	RTR	Length
1	001	0000	010	0	0000

Host acknowledges:

Dir	Group	Device	Frame Type	RTR	Length	Node ID	Slave ID
0	001	0000	000	0	0010	0010 0000	0010 0000

Host acknowledges the boot request with:

Dir = 0	Host to slave
Group = 1	Boot request response group
Device = 0	Always 0 in boot response
Frame = 0	Boot request response frame
Rtr = 0	Always 0
Length = 2	Two data bytes in return message

NOTE: Boot MID is the same for all nodes

I/D:

Node ID	Group ID (2) + Device Address (0)	" "	00h	Must resend with Group & Address
Slave ID	Same as Node ID (hex 20)	" "	00h	

Example 2. The controller is set to address 6

Controller sends:

Dir	Group	Device	Frame Type	RTR	Length
1	001	0110	010	0	0000

Host acknowledges:

Dir	Group	Device	Frame Type	RTR	Length	Node ID	Slave ID
0	001	0000	000	0	0010	0010 0110	0010 0110

Host acknowledges the boot request with:

Dir = 0	Host to slave
Group = 1	Boot request response group
Device = 0	Always 0 in boot response
Frame = 0	Boot request response frame
Rtr = 0	Always 0
Length = 2	Two data bytes in return message

NOTE*:* Boot MID is the same for all nodes

I/D:

Node ID	Group ID (2) + Device Address (6)	"&"	Hex	26
Slave ID	Same as Node ID (hex 26)		Hex	26

The controller will save the Node ID to use for message filter Group ID.

CAN Host and Controller Exchanges

When a controller receives a command, finishes a command, encounters an error condition or responds to a query, its sends an answer frame to the host using the same frame type as the command it belongs to. The answer frame format is device dependent. Generally, it will have the following format:

<MID><DLC><Answer>

Where:

<MID>: 11-bit message identifier. The direction bit is 1. The group number and the frame type are the same as received. Device is the current device address.

<DLC>: 4-bit data length code.

<Answer>: Data bytes block. The first byte of the data block is always the status byte. The second byte is a null character. The remaining bytes contain the response in ASCII format. If the reply consists of more than six bytes, then multi-frame messages are used.

NOTE: Only one command of a given frame type can be in progress at any one time: e.g., after issuing a command to a controller with frame type = 1, the host must wait for the answer with frame type = 1 before issuing the next command with frame type = 1. If the user insists on sending the command, a Command Overload status results. Several commands with different frame types can be in progress at the same time; e.g., an action command and a query command.

Following are typical exchanges between the host and controller for action commands, multi-frame commands, common commands, and query commands.

Action Command

The host commands [ZR] to a controller, and the controller is set to address 0.

Host sends:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
0	010	0000	001	0	0010	ZR

Controller Acknowledges:

Dir	Group	Device	Frame Type	RTR	DLC
1	010	0000	001	0	0000

After executing the command, controller reports status:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
1	010	0000	001	0	0010	<60h><00h>

NOTE: The mixed formats ASCII and hexadecimal are used in the data bytes block. The hexadecimal number is bracketed (< >). The rest of the fields are displayed in binary format.

Multi-Frame command

The host commands [ZM10IM10OgHIM5OM5G10G5R] to a controller, and the controller is set to address 0.

Host Sends:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
0	010	0000	011	0	1000	ZM10IM10
0	010	0000	100	0	1000	OgHIM5OM
0	010	0000	001	0	0111	5G10G5R

Controller acknowledges:

Dir	Group	Device	Frame Type	RTR	DLC
1	010	0000	001	0	0000

After executing the command, controller reports status:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
1	010	0000	001	0	0010	<60h><00h>

NOTE: For multi-frame commands, the controller only acknowledges the last frame.

Common Command

After the host has sent command [ZM10B] to the controller, it sends command 1 of frame type 2 to the controller and makes the valve move. The controller is set to address 0.

Host sends:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
0	010	0000	010	0	0001	1

Controller acknowledges:

Dir	Group	Device	Frame Type	RTR	DLC
1	010	0000	010	0	0000

After executing the command, controller reports status:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
1	010	0000	010	0	0010	<60h><00h>

Query Command

Example 1. The host sends Report command 29 of frame type 6 to a controller, and the controller is set to address 1.

Host sends:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
0	010	0001	110	0	0010	29

Controller reports:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
0	010	0001	110	0	0010	<60h><00h>

NOTE: For query commands, no acknowledge frame is needed.

Example 2. The host sends Report command 23 of frame type 6 to a controller, and the controller is set to address 1.

Host sends:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
0	010	0001	110	0	0010	23

Controller reports:

Dir	Group	Device	Frame Type	RTR	DLC	Data Bytes
1	010	0001	011	0	1000	<60h><00h>ValveC
1	010	0001	100	0	1000	ntrl:<20h>10
1	010	0001	110	0	0100	2114

NOTE: For a multi-frame Response, the start frame is type 3, the middle frame is type 4, and the last frame is type 6.

Valve Controller Command Set Description

The Valve Controller features a robust command set which allows a wide range of valve actions. Many of the commands have default arguments; however, the default values may not provide the optimal settings for your application.

When problems are detected, the Valve Controller sends an error code. The error codes are described in the Valve Controller Error Codes section.

Command Execution Guidelines

To use the commands properly, keep the following in mind:

- All commands, except Report commands and most Control commands must be followed by a [R] (Run) command.

- Single or multiple command strings can be sent to the controller.

  For example:
  - A single command such as [IR] moves the valve to the input position.

  - A multi-command string such as [IOBR] moves the valve from input position to output position and finally to the bypass position.

- The controller's command buffer holds a maximum of 96 characters. If a command is sent without the [R] (Run) command, it is placed into the buffer without being executed. If a second command is sent before the first command is executed, the second command overwrites the first command (i.e., the first command string is erased).

- Once a command is executed, new commands are not accepted until the sequence is completed. If a new command is issued while the controller is busy, an error code will be returned. Exceptions to this rule include the [T] (Terminate) and all Report commands.

- When a command is sent, the controller answers immediately. If an invalid command has been sent in a command string, the controller reports an error immediately. If there was an invalid parameter, in certain cases, the string will execute until the invalid parameter and then stop. In this case, the [Q] (Query) command can be used to read back the error to the host computer.

Command Syntax

The syntax for each command in the command set is:

[command]	Square brackets [ ] are used to distinguish commands and should not be sent as part of the command string.
<n>	Command argument. n must be a value within the specified range.
0...6000	Valid range of numerical <argument>
(n)	Default <argument>, if no argument is given
{n}	Power up default value

NOTE: All commands are case sensitive

Initialization Commands

NOTE: On power up, the TCS Valve Controller automatically homes the valve motor. This feature can be disabled by using the EEPROM configuration command [u2_0].

Initialization Sequence

The initialization sequence, in response to the [Z] or [Y] commands, or automatically on power up are as follows:

1) The Valve Controller scans 360 degrees looking for the home position.
2) The valve moves to the designated home position. On power up, it is assumed a [Y] initialization command was issued.

Z<n> Initialize Valve, Set Valve Output to the Right or CW polarity

The [Z] command initializes the valve drive and sets a non-distribution type valve output to the right (as viewed from the front of the controller).

If the controller is configured for a distribution valve, <n> selects the final port position at completion of the initialization process.

<n>	0…X	Final port position upon completion of initialization process
(n)	X	Default argument, X = number of valve ports

NOTE: X = number of ports on the valve

Argument <n>	**Description**
0	Set the final initialization port to X (default)
1…X	Set the final initialization port to the specified value

For <n> to be valid, a distribution valve must be selected. If a non-distribution valve is selected, this argument is ignored.

Valve ports are numbered 1...X starting in a clockwise direction. See valve figures below for more details.

Y<n> Initialize Valve, Set Valve Output to the Left or CCW polarity

The [Y] command initializes the valve drive and sets a non-distribution valve output to the left (as viewed from the front of the controller).

If the controller is configured for a distribution valve, <n> selects the final port position at completion of the initialization process.

<n> 0…X Final port position upon completion of initialization process
(n) X Default argument, X = number of valve ports

NOTE: X = number of ports on the valve

Argument <n>	Description
0	Set the final initialization port to X (default)
1...X	Set the final initialization port to the specified value

For <n> to be valid, a distribution valve must be selected. If a non-distribution valve is selected, this argument is ignored.

Valve ports are numbered 1...X starting in a CCW direction. See valve figures below for more details.

w<n> Initialize Valve

Same as Y[n] command

Valve Movement Commands

I Move Valve to Input position (non-distribution valves)

The [I] command moves the valve on the Valve Controller to the Input position. The Input position can be either to the Left or the Right depending on the [Y] or [Z] initialization commands.

For example:
If the [I] command is sent after the [Z] command, the valve connects the common port to the left side port (as viewed from the front of the controller).

The figures below show the positions of the valves in relation to the initialization command, valve commands, and the valve installed.

NOTE: A 4-port distribution can also be controlled using the [I], [O], [B] and [E] commands by configuring with the [U4] command. See Figure 4 for more details.

I<n> Move Valve clockwise to port <n> position (distribution valves only)

<n> 0…X Desired port position, X = number of valve ports
(1) Default argument

The [I]<n> command moves the distribution valve on a Valve Controller to the port <n> in a clockwise motion. [I0] sets valve to port 1.

A distribution valve is selected by using the U command. For example, a 3-port distribution valve is selected using [U11].

The figures below show the positions of the valves in relation to the initialization command, valve commands, and the valve installed.

O Move Valve to Output position (non-distribution valves)

The [O] command moves the valve on the Valve Controller to the Output position. The Output position can be either to the Left or the Right depending on the [Y] or [Z] initialization commands.

The figures below show the positions of the valves in relation to the initialization command, valve commands, and the valve installed.

O<n> Move Valve counterclockwise to port <n> position (distribution valves only)

<n>	0…X	Desired port position, X = number of valve ports
(X)		Default argument

The [O]<n> command moves the distribution valve on a Valve Controller to the port <n> in a counterclockwise motion. [O0] sets valve to port X.

A distribution valve is selected by using the U command. For example, a 3-port distribution valve is selected using [U11].

The figures below show the positions of the valves in relation to the initialization command, valve commands, and the valve installed.

B Move Valve to Bypass position (non-distribution valves)

For a 3-port Y valve:
The [B] command connects the Input and Output positions, bypassing the common port.
For a 3-port/4-port T valve, Loop valve, or 4-port Distribution valve using IOBE:
The [B] command moves the valve to the Extra position as specified in the figures below.

E Move Valve to the Extra position

For a 3-port Y valve:
Same as the [B] Bypass command.

For a 90°, 4-port, non-distribution valve:
The [E] command moves the valve to the extra position.

For a 3-port/4-port T valve, Loop valve, or 4-port Distribution valve using IOBE:
The [E] command moves the valve to the extra position as specified in the figures below.

For Distribution valves:
The [E] command moves the valve to the desired port using the shortest distance in either the CW or CCW direction.
NOTE: Same as the [A] command.

A<n> Move Valve to port <n> position using the shortest path (distribution valves only)

<n>	0…X	Desired port position, X = number of valve ports
(1)		Default argument

The [A]<n> command moves the distribution valve on a Valve Controller to the port <n> in a clockwise (CW) or counterclockwise (CCW) motion. The direction of rotation is whichever allow the shortest move distance. [A]<0> sets valve to port 1.

a<n> Move Valve to port <n> position using the shortest path (distribution valves only)

Same as A[n].

Figure 1. Valve Positions for the 3-Port, 120°, Y Valve (U1 set)

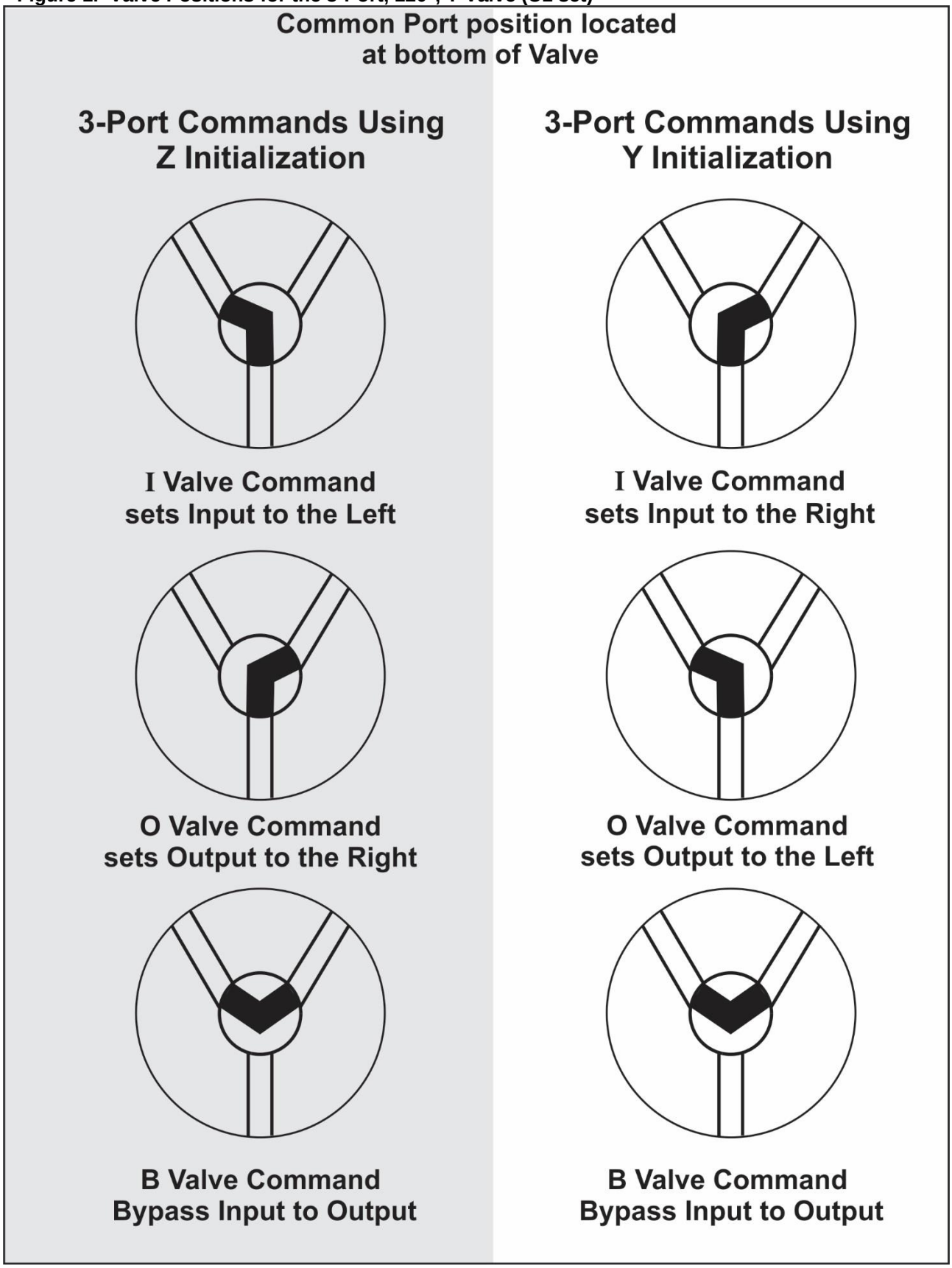


Figure 2. Valve Positions for the 4-Port, 90°, Valve (U2 set)

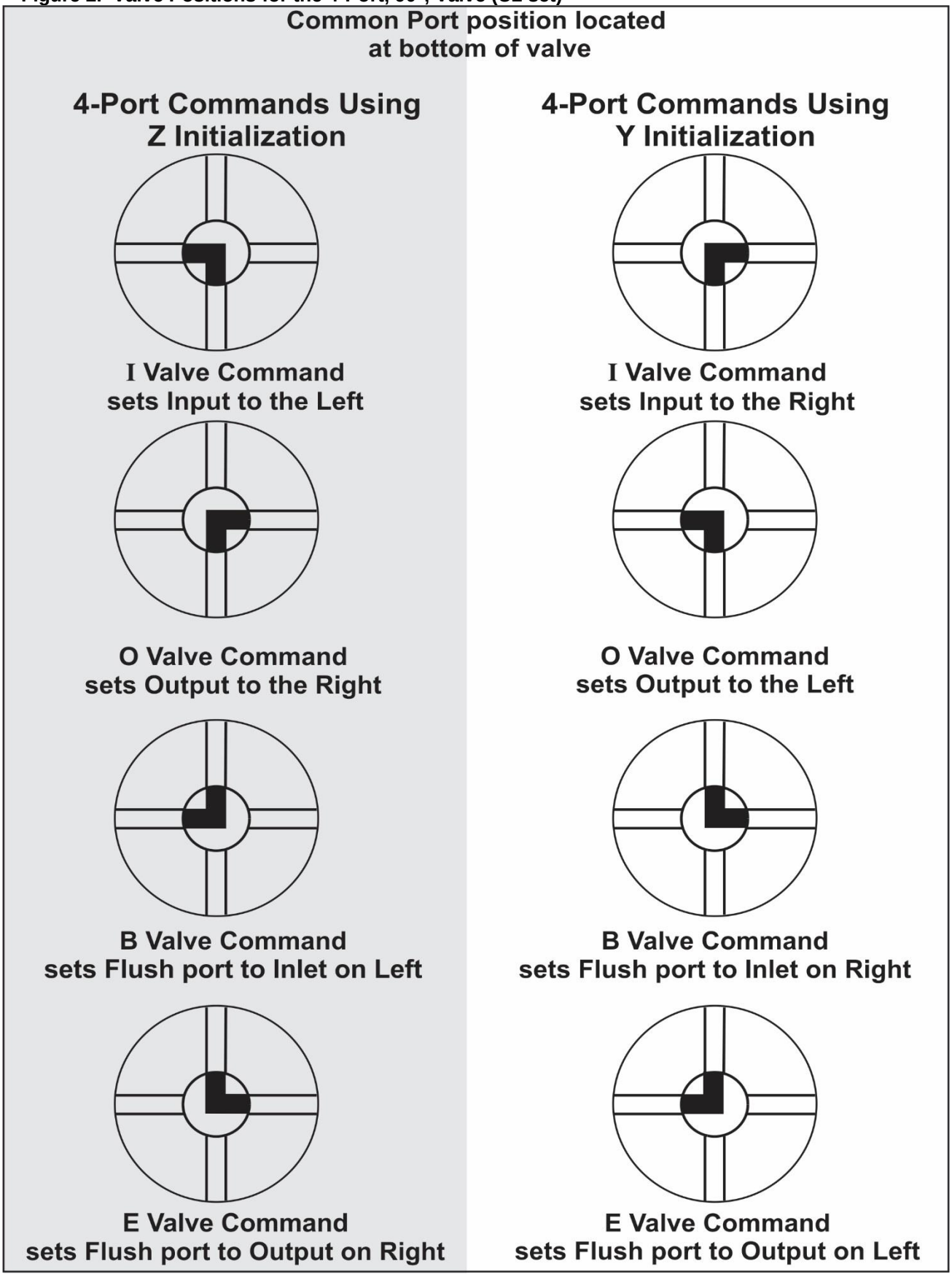


Figure 3. Valve Positions for the 3-Port or 4-Port, 90°, T Valves (U5 set)

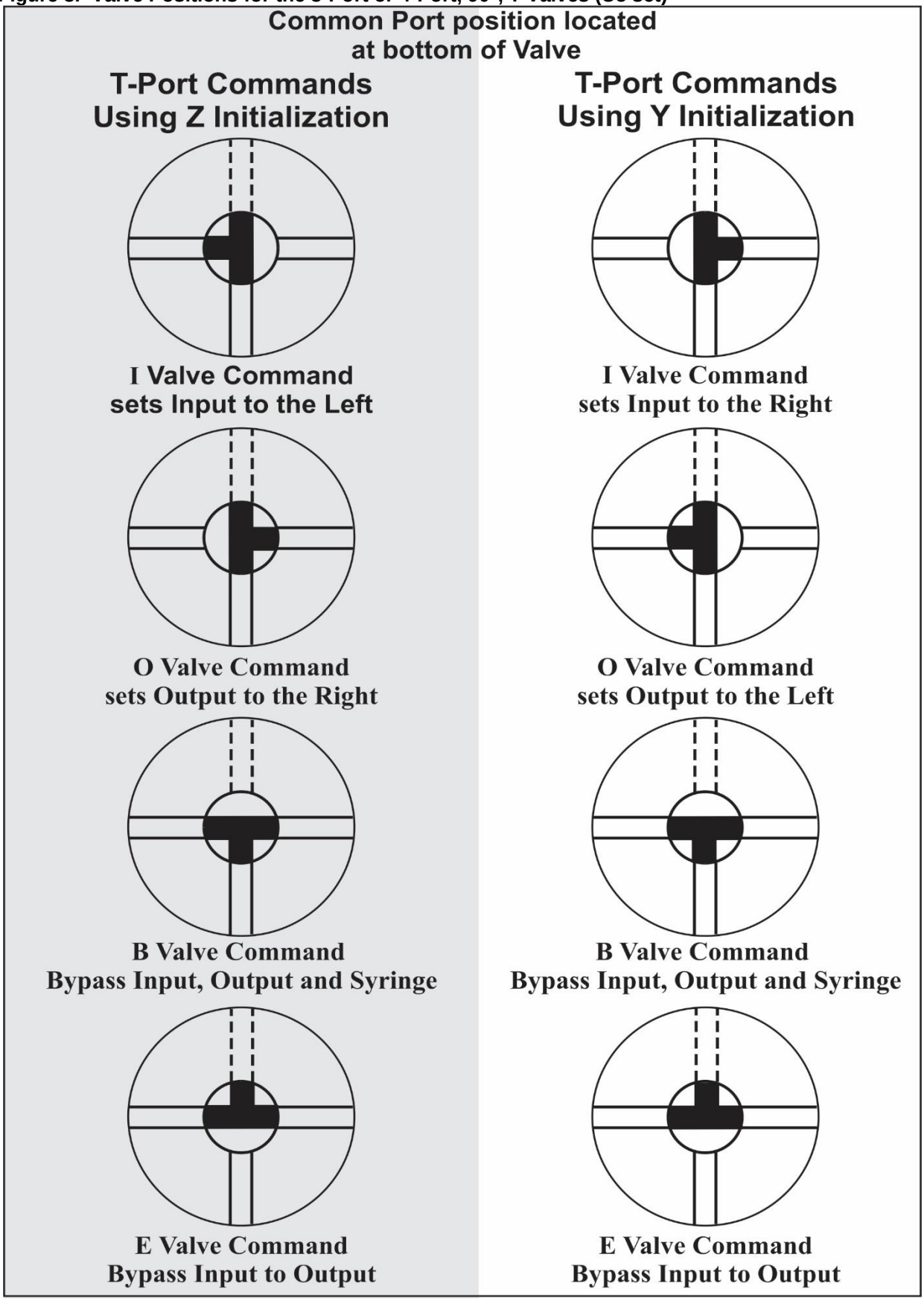


Figure 4. Valve Positions for the 4-Port Distribution Valve, using IOBE (U4 set)

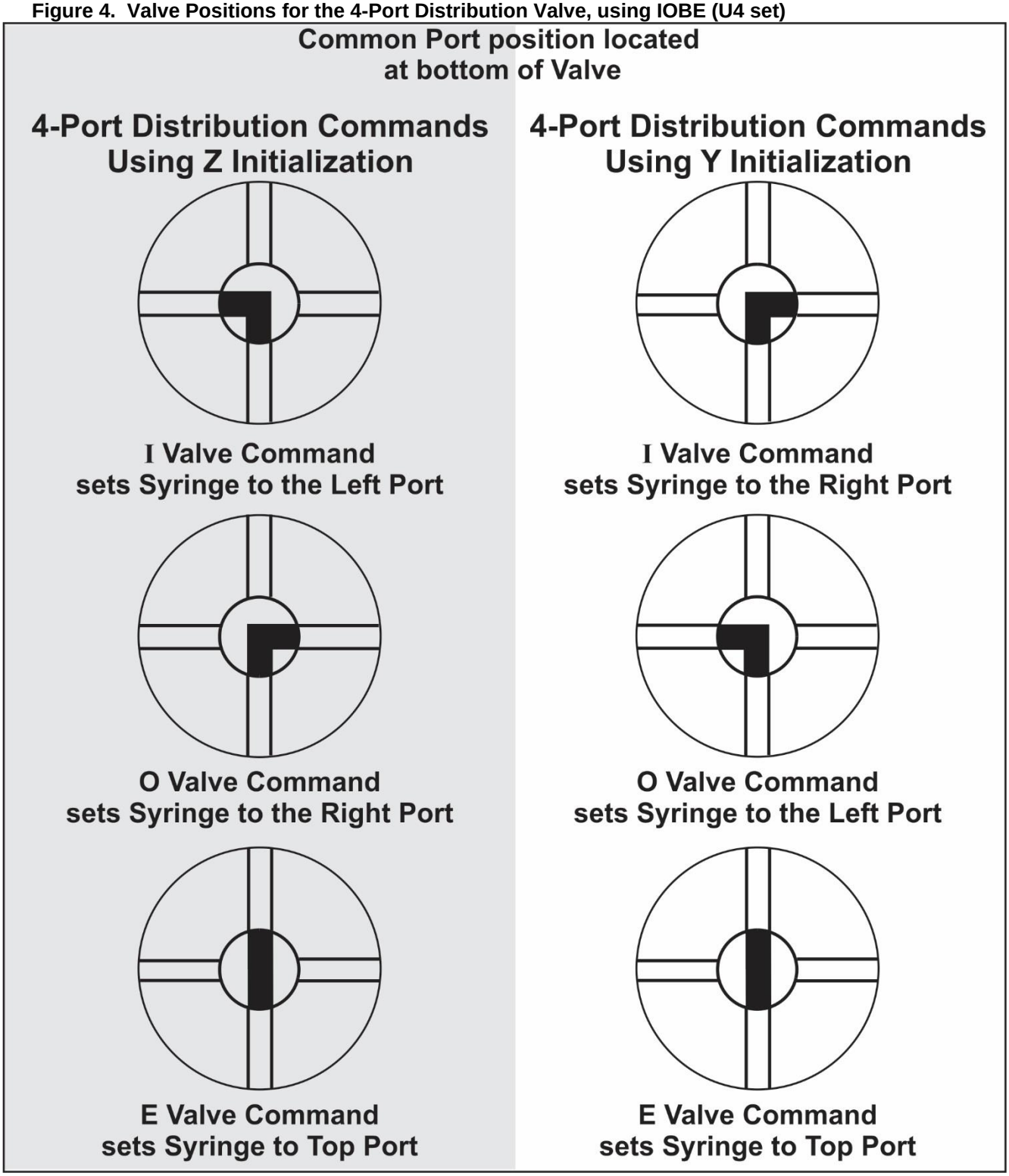


Figure 5. Valve Positions for the 4-Port Distribution Valve (U11 set)

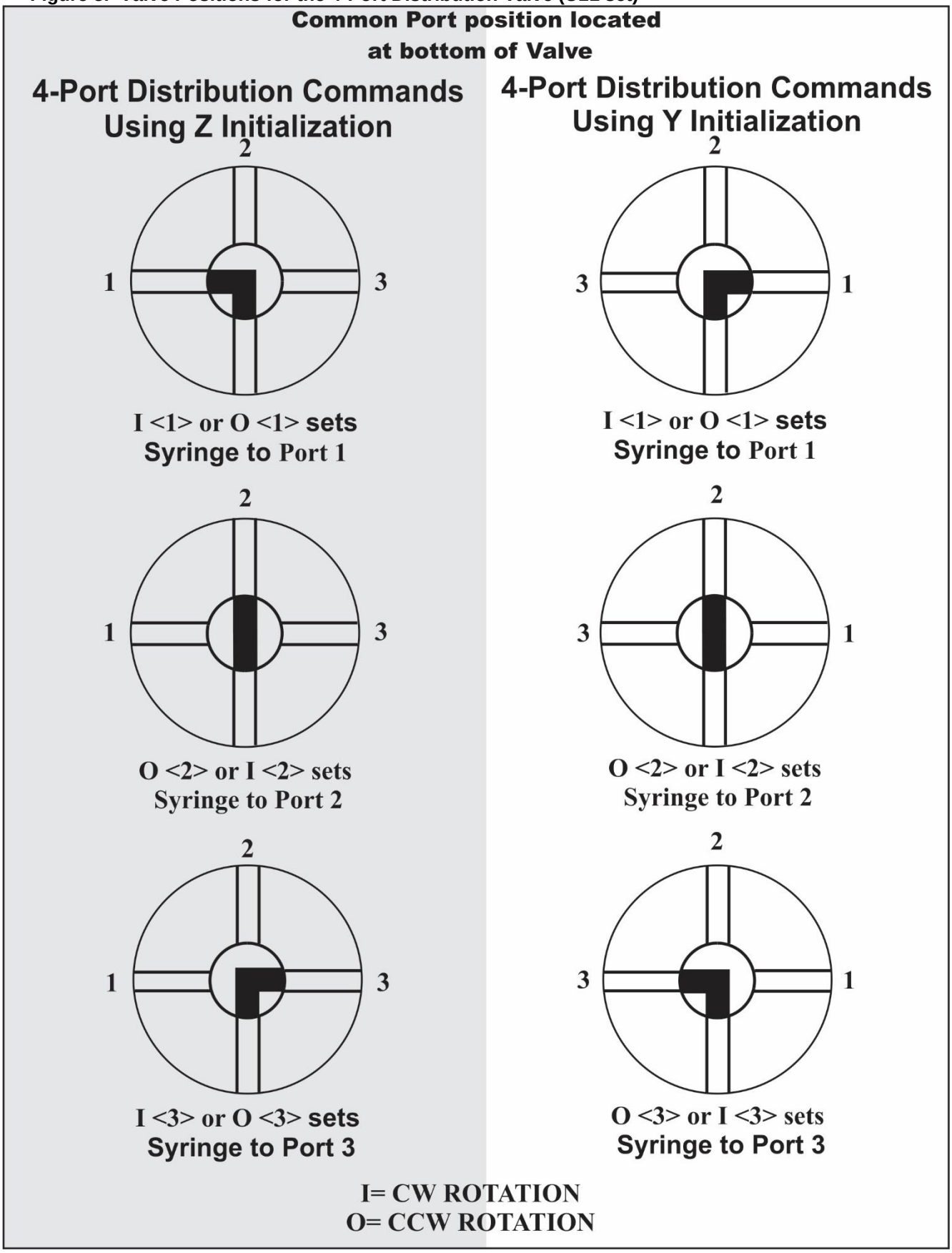


Figure 6. Valve Positions for the 4-Port Loop Valve (U9 set)

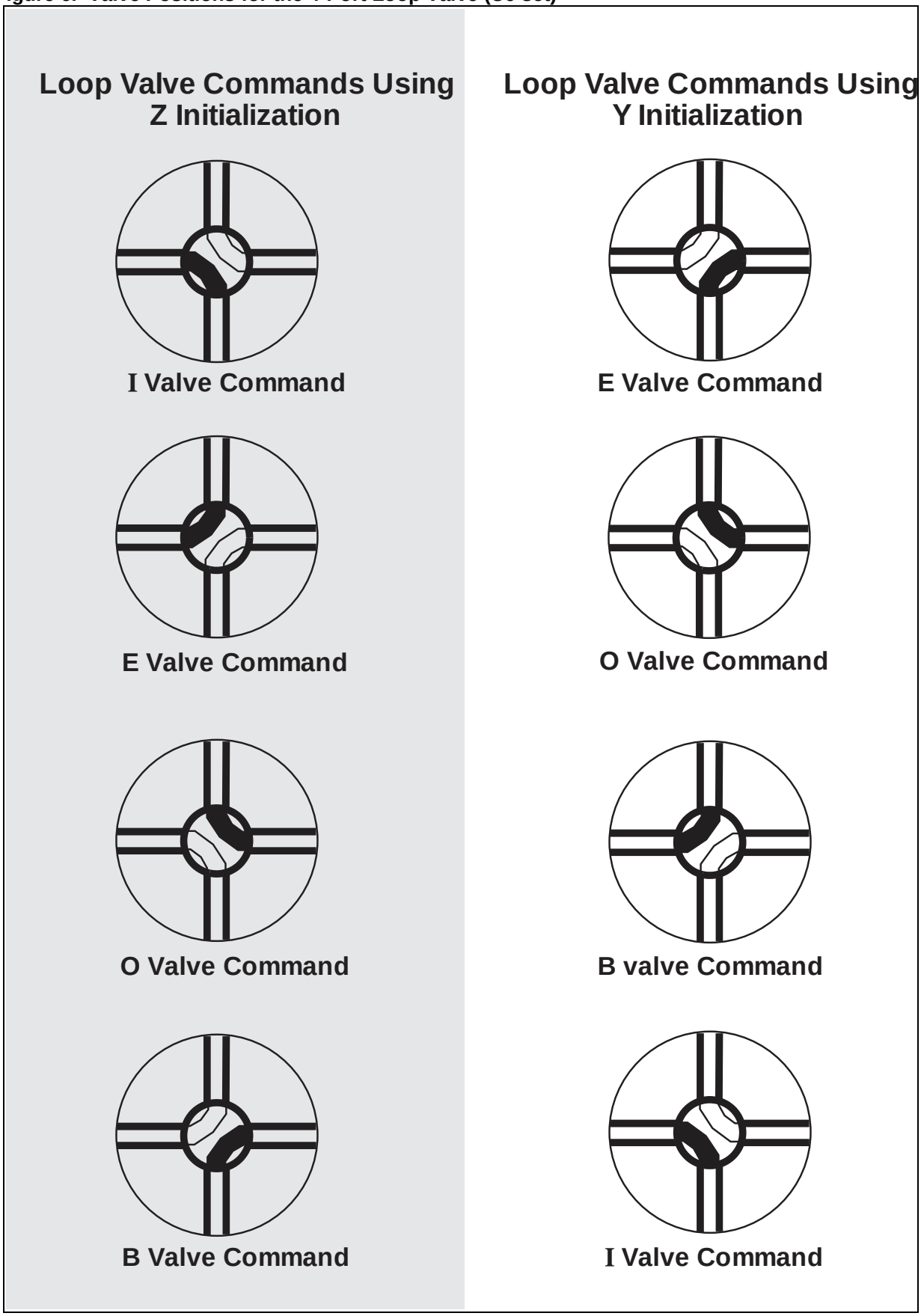
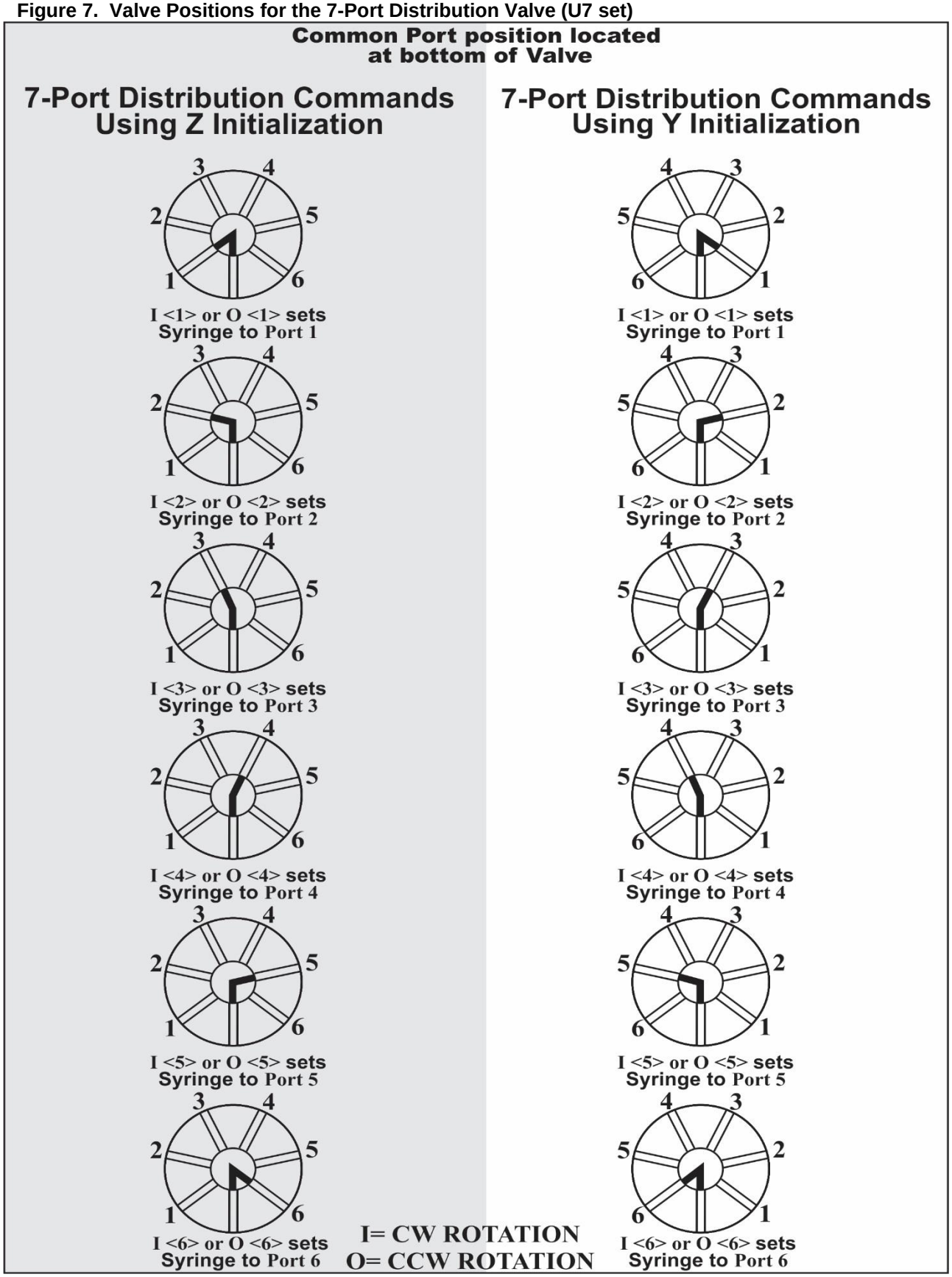


Figure 7. Valve Positions for the 7-Port Distribution Valve (U7 set)



Set Commands

Set commands are used to configure the controller for specific operations.

J<n> Auxiliary Outputs

The [J] command sets the three auxiliary TTL output lines.

The syntax for this command is:

<n> 0...7
{0} Power up default

The Valve Controller provides three TTL outputs on connector JP3 (pins 13, 14, 15) that correspond to outputs 1, 2, and 3. They are controlled as follows:

Command	Output 3 (pin 15)	Output 2 (pin 14)	Output 1 (pin 13)
J0	0	0	0
J1	0	0	1
J2	0	1	0
J3	0	1	1
J4	1	0	0
J5	1	0	1
J6	1	1	0
J7	1	1	1

(0 = Low, Gnd; 1 = High, +5VDC)

Control Commands

R Execute Command or Program String

The [R] command tells the controller to execute a new or previously loaded but unexecuted command string. This command will also cause the resumption of a halted [H] command string.

Commands containing [R] at the end of the string will execute immediately. If the command or program string is sent without the [R], it is placed in the command buffer.

Sending the [R] alone will execute the last unexecuted command in the buffer. Sending another [R] will not repeat the program string that has previously been executed.

X Execute the Last Command or Program String

Repeats the last executed command string. Note that strings that contain looping commands, [g] and [G] are not valid when using [X].

G<n> Repeat Command Sequence

This command repeats a command or program string the specified number of times. If a [GR] or a [G0R] is sent, the sequence is repeated indefinitely or until a [T] terminate command is issued. The [G] command can be used to nest up to 10 loops and can be repeated up to 30,000 times.

<n>	0...30000	# of times to repeat
(0)		Default argument

NOTE: The argument <0> is a special case. It causes the loop to repeat indefinitely.

For example:
[IM1000OM1000G10R] moves the valve to the Input position, waits 1000 mS, moves to the Output position, and finally waits 1000 mS. This sequence is repeated 10 times.

g Mark the Start of a Repeat Sequence

The [g] command is used in conjunction with the [G] command. The [g] command marks the beginning of a repeat sequence (loop) that occurs within program string (i.e., the entire string is not repeated). Both the [g] and [G] commands can be used to nest up to 10 loops.

M<n> Delay Command Execution

The [M] command delays execution of a command in milliseconds. The syntax for this command is:

<n> 0...30000 milliseconds

H<n> Halt Command Execution

The [H] command is used within a program string to halt execution of the string. To resume execution, a [R] command or TTL signal must be sent.

<n>	0...2
(0)	Default argument

The [H] command is used within a program string to halt execution of the string. To resume execution, a [R] command or TTL signal must be sent. Two TTL inputs are available, Input 1 (JP3 pin 7), and Input 2 (JP3 pin 8). Both inputs are pulled up to +5V on the controller.

Halt Command execution is as follows:
<n> = 0 Waits for [R], or Low signal on either Input 1 or 2
<n> = 1 Waits for [R], or Low signal on Input 1
<n> = 2 Waits for [R], or Low signal on Input 2

Note that the inputs are level sensitive. That is, after the [H] command, if the respective Input is low, operation will continue. The input does not need to transition from High to Low.

For example, the following command string is sent:

[ZgH1IH2OG0R]

The controller initializes, waits for a low signal on Input 1, then moves the valve to the Input position. It then waits for a low on Input 2 before moving the valve to the Output position:

Z Initialize controller
g Start loop
H1 Wait for low on Input1
I Valve to Input position
H2 Wait for low on Input2
O Valve to Output position
G0 Loop back to [g], run forever
R Run

The status of the TTL input lines can also be read using [?13] and [?14]. These commands are described in Report Commands section.

T Terminate Command

The [T] command immediately terminates any executing command string. Valve movements are allowed to complete before termination.

x Execute next Command in buffer based on Auxiliary inputs

Execute the next command in the string based on the Auxiliary inputs.

<n> 0...3

<n>	**Input2**	**Input1**	**Action**
<0>	Low	Low	Will execute next command in the string buffer if Input2 and Input1 are held Low
<1>	Low	High	Will execute next command in the string buffer if Input2 and Input1 are set as indicated
<2>	High	Low	Will execute next command in the string buffer if Input2 and Input1 are set as indicated
<3>	High	High	Will execute next command in the string buffer if Input2 and Input1 are both High

Input 1 is located on connector JP3 pin 7, and Input 2 on JP3 pin 8.

NOTE: Auxiliary inputs are internally pulled High. If they are left unconnected, they will be High.

The [x] command in conjunction with the [e] command can be used as an IF..ELSE, IF..ELSE statement based on the auxiliary TTL inputs. This is especially useful in stand-alone operation.

The example below allows the Valve Controller in standalone mode to switch between 4 different valve positions based on the auxiliary inputs.

If both inputs are low, [e1] will be executed. The controller will move the valve to the Input position.
If Input 2 is Low and Input 1 is High, [e2] will be executed. The controller will move the valve to the Output position.
If Input 2 is High and Input 1 is Low, [e3] will be executed. The controller will move the valve to the Bypass position.
If both inputs are High, [e4] will be executed. The controller will move the valve to the Extra position.

The following four strings are programmed into the EEPROM locations 0, 1, 2, 3 and 4:

[s0Zx0e1x1e2x2e3x3e4R]
[s1IR]
[s2OR]
[s3BR]
[s4ER]

Non-Volatile Memory (EEPROM) Commands

The non-volatile memory in the Valve Controller can store up to 15 separate program strings. This provides the user with the option of computer-free standalone operation. The controller can be programmed to run stored programs using the [s] command, the AutoRun switch and rotary address switch.

On power up, if the AutoRun switch, SW2#1, is in the Off position, the string corresponding to the rotary switch position will be automatically executed.

For example:
The following 2 strings are programmed into locations 0 and 1:
[s0ZIH0OR]
[s1ZOH0IR]

Put the AutoRun switch, SW2#1, in the Off position.
If the rotary switch on the controller is set to 0, [s0] will run automatically. If the rotary switch is set to 1, [s1] will run automatically.

s<n> Load Program String into Non-Volatile Memory

The non-volatile memory in the Valve Controller can store a program string providing the user with the option of computer-free operation. The [s] command is placed at the beginning of a program string to load the string into the non-volatile memory.

<n> 0...14

Up to 15 program strings (numbered 0 through 14) can be loaded into the non-volatile memory. Each string can use up to 128 characters. For example, [IM3000OM10R] requires 11 characters.

To run a stored string automatically on power up, the AutoRun s must be installed and the rotary address switch set to the proper address. The table below shows the relationship between the stored strings [s<n>] and the rotary switch.

Rotary Switch Setting	String Referenced
0	s0
1	s1
2	s2
3	s3
4	s4
5	s5
6	s6
7	s7
8	s8
9	s9
A	s10
B	s11
C	s12
D	s13
E	s14

The [e<n>] command is used to execute the string.

NOTE: An Initialization command should always be included in the non-volatile memory command string if the controller will be used in standalone mode.

Example program string: [s8ZgIH0OGR]

s8	Loads string into the program 8 of non-volatile memory (Address switch position 8)
Z	Initializes controller
g	Marks start of loop
I	Valve to Input position
H0	Halt operation until either TTL inputs go Low
O	Valve to Output position
G	Go to [g], repeat indefinitely
R	Executes command string

e<n> Execute Non-Volatile Memory Program String

Execute string stored at location <n>.

<n> 0...14

For example:
[e8R] will run the string stored in EEPROM location 8.

Linking Program Strings in the Non-Volatile Memory

Non-volatile memory program strings can be linked by ending one program string with an [e] command that refers to a second program string.

Example program strings: [s1Ze2R]
[s2IH0OGR]

The first string loads an initialization sequence into program 1 of the non-volatile memory (address switch position 1). It then links to string 2 in the non-volatile memory.

NOTE: When linking program strings, a jump to string will not return to the calling string.

In the example above: [s1] jumps to [e2]. Once [e2] has executed, control will **not** return to the calling string [s1]. Thus, [e] commands are normally placed at the end of a string just before the [R].

u<n_xx> Set Controller Configuration EEPROM Parameters

For factory use only.

Will load controller configuration and calibration info into the internal EEPROM. Note that, these parameters are only read on power up, and will only take effect when the power is cycled. This command, unlike the other Set commands, does not require an [R] to execute. The EEPROM settings can be reported by using the [?27] command.

u1_XX XX = Valve type
u2_X Auto initialize valve; 1 = auto init, 0 = no init on power-up
u3_XX XX = CAN communication rate

U<n> Set Controller Configuration EEPROM Parameters

Note, these parameters are only read on power up. Thus they will only take effect when the power is cycled. This command, unlike the other Set commands, does not require and [R] to execute.

<n>	**Configuration Setting**
1	3-port Y valve
2	4-port valve
4	4-port Distribution valve using [I],[O],[B],[E] commands
5	3-port/4-port T valve.
6	6-port Distribution valve
7	7-port Distribution valve
9	4-port Loop valve
11	4-port Distribution valve using [I]<n> and [O]<n> commands
51	Set CAN baud rate to 100K
52	Set CAN baud rate to 250K
53	Set CAN baud rate to 500K
54	Set CAN baud rate to 1Meg
57	Set CAN baud rate to 125K

Report Commands

Report commands report various controller parameters. The response is returned immediately and can be used when the controller is busy executing another command string.

Report commands do not require a [R] command.

? or ?6 Report Valve Position

Reports the valve position in mnemonics (i = input, o = output, b = bypass and e = extra). If a distribution valve is selected, returns the port number 1...X, where X is the number of ports.

?1, ?2, ?3 Non-functional

Always reports 900

?4 or ?5 Report Valve Position

Same as ?

?8 Report Opto state

0 = Opto blocked
1 = Opto open

?9 Non-functional

Always reports 20

?10 or F Report Command Buffer Status

Reports the command buffer status. If the buffer is empty, the controller returns status code 0. If the buffer is not empty, the controller returns a 1. If a program string is sent to the controller without a [R] command, the string is loaded into the buffer and the buffer status becomes 1. A [R] command will then execute the command stored in the buffer.

0 = empty
1 = commands in buffer

?13 Report Status of Auxiliary Input #1 (JP3, Pin 7)

0 = Low
1 = High

?14 Report Status of Auxiliary Input #2 (JP3, Pin 8)

0 = Low
1 = High

?15 Non-functional

Always reports 1

?18 or % Report Number of Valve movements since last reported

Reports the number of valve movements since last reported. Every time this command is used, the valve movement count is reset to zero.

?19 **Report if Controller has been initialized**

0 = not initialized
1 = initialized

?20 or # Report Firmware Checksum

Reports back the firmware checksum

?22 Non-functional

Always reports 255

?23 or & Report Firmware Version

Reports the firmware version. Format is "ValveCntrl: MMDDYY".

?26 Non-functional

Reports 239

?27 Report Configuration EEPROM

Reports the confirmation EEPROM data set by the [u] command

?29 or Q Report System Status

Reports error codes and controller status

?30 - ?44 Report user programmed EEPROM strings

[?30] reports [s0] string
[?31] reports [s1] string
[?32] reports [s2] string
[?33] reports [s3] string
[?34] reports [s4] string
[?35] reports [s5] string
[?36] reports [s6] string
[?37] reports [s7] string
[?38] reports [s8] string
[?39] reports [s9] string
[?40] reports [s10] string
[?41] reports [s11] string
[?42] reports [s12] string
[?43] reports [s13] string
[?44] reports [s14] string

?76 Reports Valve Controller Configuration

Example:
4DIST-IOBE/9600/100K/AUTOINIT-ON

4DIST-IOE = 4-port Distribution valve using IOBE commands
9600 = 9600 baud RS232/RS485 communication rate
100K = 100K CAN bus communication rate
AUTOINIT-ON = Auto initialize valve on power up is enabled

Valve Controller Status and Error Codes

The [Q] command reports error codes and valve controller status (Idle or Busy). The user should send a [Q] command before sending a program string or individual command to ensure that the controller has completed the previous command successfully.
The response to the [Q] command (the status byte) provides two items of information: Controller status (bit 5) and error code (bits 0-3).

Status Bit

Bit 5 is the status bit. It indicates when the controller is busy or not busy. The designations for bit 5 are listed below.

Status Bit 5 Description

X = 1	Controller is Idle. It is ready to accept new commands.
X = 0	Controller is Busy and will only accept Report or Terminate (T) commands.

Commands addressed to multiple controllers at once cannot be used to obtain controller status. Each controller must be queried separately.

NOTE: Although the answer block for other commands contains a status bit, it should not be used for determining controller status. A [Q] command is the only valid method to determine if the controller is busy. The error information in the status byte of the answer block is always valid.

Error Codes

Error codes describe problem conditions that may be detected by the Valve Controller. Error codes are returned in the least significant four bits of the status byte. If an error occurs, the controller stops executing commands, clears the command buffer, and inserts the error code into the status byte for the next Query [Q] command.

Table 5. Valve Controller Error Codes

Error Code	Status Byte if Idle (ACSII)	Status Byte if Busy (ASCII)	Description
0 (00h)	`	@	**Error Free Condition**
1 (01h)	a	A	**Initialization Error** This error occurs when the controller fails to initialize. The controller will not accept commands until it has been successfully initialized. This error can only be cleared by successfully initializing the controller.
2 (02h)	b	B	**Invalid Command** This error occurs when an unrecognized command is issued. Correct the command and operation will continue normally.
3 (03h)	c	C	**Invalid Operand** This error occurs when an invalid parameter (<n>) is given with a command. Correct the parameter and controller operation will continue normally.
4 (04h)	d	D	**Invalid CheckSum** In OEM mode, the checksum did not match the received string.
6 (06h)	f	F	**EEPROM Failure** This error occurs when the EEPROM is faulty. If you receive this error, please call TriContinent Customer Service.
8 (08h)	h	H	**CAN Bus Failure**
10 (0Ah)	j	J	**Valve Overload** This error occurs when the valve drive loses steps and can't position itself correctly. The controller must be reinitialized before normal operation can resume. Sending another Valve motion command automatically reinitializes the valve and sets it to the correct location. Continual valve overload errors could be an indication that the valve should be replaced.
15 (0Fh)	o	O	**Command Overflow** This error occurs when the command buffer contains more than 96 characters. Commands in the buffer must be executed before more commands can be sent. A command overflow will also occur if the controller is busy executing a command and another command string is requested to run.

The controller handles errors differently depending on the error type. There are four error types, which are described below.

Immediate Errors (error 2, 3, and 4)

These include "Invalid Command" (error 2), "Invalid Operand" (error 3), and "Invalid Checksum" (error 4). After the command is sent, the answer block immediately returns an error. Once a valid command is sent, the controller will continue to function normally. Since the [Q] command is a valid command, the controller will not return an error. In this case, the [Q] command is not required.

There is no need to reinitialize the controller following this error type.

Initialization Error (error 1)

To ensure that the valve controller initializes successfully, after issuing the [Z], [Y] or [w] command, poll the controller with the [Q] command to indicate a successful initialization. If the valve controller is set to automatically initialize on power-up (factory default), the [Q] command should be used to determine if an initialization error occurred on power-up.

- If an error occurs during the initialization, an "Initialization Error" (error 1) will be returned with the [Q]'s response. The controller should be reinitialized with the [Z], [Y] or [w] command, and then polled with the [Q] command to indicate a successful initialization.

- If the [Q] command indicates both a successful initialization and that the controller is ready, subsequent valve Move commands can be sent.

- If Initialization is not successful, any subsequent valve move commands ([I], [O], [B], [E], [A], [a]) will automatically attempt to reinitialize the valve before positioning to the commanded valve position.

Valve Overload Error (error 10)

If the controller returns a "Valve Overload" (error 10), the controller must be reinitialized before continuing. There are two ways to reinitialize:

1. Send a valve initialization command ([Z], [Y] or [w]).
2. Any valve move commands ([I], [O], [B], [E], [A], [a]) will automatically attempt to reinitialize the valve.

Command Overflow Error (error 15)

This error occurs when certain commands are sent to the controller while it is busy executing a previous command string. Any Valve commands that are sent while the controller is busy, will cause this error to be issued. The controller ignores the command and issues a "Command Overflow" (error 15).

The [Q] command, based on the "Busy or "Idle" response, allows the Host to determine when the command, or command string, is completed. Once the valve controller response is "Idle", it is ready to accept new commands.
There is no need to reinitialize the controller following this error type.

Report commands and the terminate [T] command will not return a "Command Overflow" error. These commands are considered valid even when the controller is Busy.

Valve Controller Hardware Faults (error 6 and 8)

These indicate a fault with the valve controller hardware. The unit should be returned for servicing. Please contact TriContinent Customer Service if these errors occur.

Table 6. Error Codes, Status Byte with ASCII and Hexadecimal Values

Status Byte	**Hex if Status Bit 5 =**		**ASCII if Status Bit 5 =**		**Error Code**	**Description**
7 6 5 4 3 2 1 0	**X= 0 (Busy)**	**X=1 (Idle)**	**X= 0 (Busy)**	**X=1 (Idle)**	**Number**	**Error**
0 1 X 0 0 0 0 0	40h	60h	@	`	0	No Error
0 1 X 0 0 0 0 1	41h	61h	A	a	1	Initialization Error
0 1 X 0 0 0 1 0	42h	62h	B	b	2	Invalid Command
0 1 X 0 0 0 1 1	43h	63h	C	c	3	Invalid Operand
0 1 X 0 0 1 0 0	44h	64h	D	d	4	Invalid Checksum
0 1 X 0 0 1 0 1	45h	65h	E	e	5	Unused
0 1 X 0 0 1 1 0	46h	66h	F	f	6	EEPROM Failure
0 1 X 0 1 0 0 0	48h	68h	H	h	8	CAN Bus Failure
0 1 X 0 1 0 1 0	4Ah	6Ah	J	j	10	Valve Overload
0 1 X 0 1 1 1 1	4Fh	6Fh	O	O	15	Command Overflow

Document History

Date	Changes
10/21/2014	• Initial version
3/31/2015	• Added information for DIP Switch settings (configuration, valve type, bus terminations)
11/16/15	• Updated cover page with new company logo. Corrected errors on pages 1 & 36. Add company locations on page 46.
4/8/16	• Revise loop valve diagram, Figure 6.

TCS Valve Controller RS232 Command Summary

Command	Operand Range <n>	Default Operand	Power up default	Command Description	Operand Description
Valve Initialization Commands					
Z<n>	0...X	0		Initialize valve to the right	X = Port (distribution valves only)
Y<n>	0...X	0		Initialize valve to the left	X = Port (distribution valves only)
w<n>	0...X	0		Same as Y<n>	X = Port (distribution valves only)
Valve Commands					
I<n>	0...X	1		Move valve to Input position	X = Port (distribution valves only)
O<n>	0...X	1		Move valve to Output position	X = Port (distribution valves only)
B<n>	0...X	1		Move valve to the Bypass position	X = Port (distribution valves only)
E<n>	0...X	1		Move valve to the Extra position	X = Port (distribution valves only)
A<n>	0...X	1		Move distribution valve to the specified position	X = Port
Control Commands					
R	N/A	N/A		Execute command string	
X	N/A	N/A		Re-execute last executed command string	
M<n>	0...30000			Delay command execution	Milliseconds
G<n>	0...30000	0		Repeat command sequence	0 = Loop forever
g	N/A	N/A		Mark the start of a repeat sequence	
T	N/A	N/A		Terminate command	
H<n>	0-2	0		Halt command execution	<0> = Wait for [R] or either input 1 or 2 to go low <1> = Wait for [R] or input 1 to go low <2> = Wait for [R] or input 2 to go low
x	0...3	N/A		Execute next command in buffer based on auxiliary inputs	<0> = Execute if Input2 is Low and Input1 is Low <1> = Execute if Input2 is Low and Input1 is High <2> = Execute if Input2 is High and Input1 is Low <3> = Execute if Input2 is High and Input1 is High
Set Commands					
J<n>	0...7			Sets the 3 TTL auxiliary outputs	0 = all outputs Low 7 = all outputs High
EEPROM Commands					
s<n>	0...14			Load program string into EEPROM	
e<n>	0...14			Execute EEPROM string	
U				Set controller configuration parameters	
u	0...3			Set system configuration parameter into EEPROM	For factory use only

Command	Operand Range <n>	Default operand	Power up default	Command Description	Operand Description
Report Commands					
Q	N/A	N/A		Report system status	
?	N/A	N/A		Reports Valve Position (i, o, b and e; or port number for distribution valve)	
?0	N/A	N/A		Same as ?	
?1	N/A	N/A		Non-Functional, will always return 900	
?2	N/A	N/A		Non-Functional, will always return 900	
?3	N/A	N/A		Non-Functional, will always return 900	
?6	N/A	N/A		Same as ?	
?9	N/A	N/A		Non-Functional, will always return 20	
?10	N/A	N/A		Report Command Buffer Status, same as F	
?13	N/A	N/A		Report Status of auxiliary 1 input	
?14	N/A	N/A		Report Status of auxiliary 2 input	
?15	N/A	N/A		Non-Functional, will always report 1	
?17	N/A	N/A		Non-Functional, will always report 1	
?18	N/A	N/A		Number of valve movements since last ?18, same as %	
?19	N/A	N/A		Reports if the valve controller was initialized 0 = not initialized 1 = initialized	
?20	N/A	N/A		Report firmware checksum, same as #	
?22	N/A	N/A		Non-Functional, will always return 255	
?23	N/A	N/A		Report firmware version, same as &	
?26	N/A	N/A		Non-Functional, will always return 239	
?27	N/A	N/A		Reports configuration EEPROM data as set using the [u] command	
?29	N/A	N/A		Current status, same as [Q] command	
?30 - ?44				Reports user EEPROM execution strings, ?30 = s0, ?31 = s1 and so on	
?76	N/A	N/A		Reports valve controller configuration in ASCII	
F	N/A	N/A		Report command buffer status, same as ?10	
&	N/A	N/A		Report firmware version, same as ?23	
#	N/A	N/A		Report firmware checksum, same as ?20	
%	N/A	N/A		Number of valve movements since last report, same as ?18	

TCS Valve Controller CAN Bus Command Summary

Command	Operands	Command Description
On-the fly Commands Frame Type = 0		
T	N/A	Terminate
Action Commands Frame Type = 1		
		All RS232/RS485 commands, with the exception of Report commands, are valid Action commands in CAN bus mode.
Common Commands Frame Type = 2		
0	N/A	Reset Boot Request mode
1	N/A	Start Loaded command, like R
2	N/A	Clear Loaded command
3	N/A	Repeat last command, like X
4	N/A	Stop action immediately, same as T command
Report Commands Frame Type = 4		
0	N/A	Reports valve position
1	N/A	Nonfunctional, always returns 900
2	N/A	Nonfunctional, always returns 900
3	N/A	Nonfunctional, always returns 900
10	N/A	Buffer status, like F
13	N/A	Input 1 status, like ?13
14	N/A	Input 2 status, like ?14
15	N/A	Nonfunctional, always returns 1
17	N/A	Nonfunctional, always returns 1
18	N/A	Number of valve movements since last report, like ?18
19	N/A	Report if controller is initialized. 1 = initialized, 0 = not initialized
20	N/A	Firmware checksum, like ?20
22	N/A	Nonfunctional, always returns 255
23	N/A	Firmware version, like &
29	N/A	Current status, like Q

TRICONTINENT
by Gardner Denver

TriContinent Scientific, Inc.
12555 Loma Rica Drive
Grass Valley, CA 95945
USA
Tel: 800-937-4738
Fax: 530-273-2586
liquidhandling.tcs@gardnerdenver.com

Gardner Denver Thomas GmbH
Livry-Gargan-Str. 10
82256 Fürstenfeldbruck
Germany
Tel: +49 8141 2280 0
Fax: +49 8141 8892136
thomas.de@gardnerdenver.com

Gardner Denver Hong Kong, Limited
Unit 1317-1318 Delta House, 3 On Yiu Street
Siu Lek Yuen, Shatin, New Territories
Hong Kong
Tel: +852 26903502
Fax: +852 27924598
thomas.hk@gardnerdenver.com

Printed in USA Form No. 8694-22 Rev. A © Gardner Denver TriContinent. All rights reserved.